TU Clausthal

Clausthal University of Technology

# Relational Topographic Maps

**Barbara Hammer and Alexander Hasenfuss**

**IfI Technical Report Series** **IfI-07-01**





Department of Informatics
Clausthal University of Technology

## Impressum

**Publisher:** Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
**Editor of the series:** Jürgen Dix
**Technical editor:** Wojciech Jamroga
**Contact:** wjamroga@in.tu-clausthal.de
**URL:** http://www.in.tu-clausthal.de/forschung/technical-reports/
**ISSN:** 1860-8477

## The IfI Review Board

# Relational Topographic Maps

Barbara Hammer and Alexander Hasenfuss

Clausthal University of Technology - Institute of Computer Science
Julius Albert Strasse 4, 38678 Clausthal-Zellerfeld - Germany

**Abstract**

We introduce relational variants of neural topographic maps including the self-organizing map and neural gas, which allow clustering and visualization of data given in terms of a pairwise similarity or dissimilarity matrix. It is assumed that this matrix originates from an euclidean distance or dot product, respectively, however, the underlying embedding of points is unknown. One can equivalently formulate batch optimization for topographic map formation in terms of the given similarities or dissimialities, respectively, thus providing a way to transfer batch optimization to relational data. For this procedure, convergence is guaranteed and extensions such as the integration of label information can readily be extended to this framework.

## 1 Introduction

Topographic maps such as the self-organizing map (SOM) constitute a valuable tool for robust data inspection and data visualization which has been applied in diverse areas such as telecommunication, robotics, bioinformatics, business, etc. [19]. Alternative methods such as neural gas (NG) [23] provide an efficient clustering of data without fixing a prior lattice. This way, subsequent visualization such as multidimensional scaling, e.g. Sammon's mapping [21, 32] can readily be applied, whereby no prior restriction of a fixed lattice structure as for SOM is necessary and the risk of topographic errors is minimized. For NG, an optimum (nonregular) data topology is induced such that browsing in a neighborhood becomes directly possible [24].

In the last years, a variety of extensions of these methods has been proposed to deal with more general data structures. This accounts for the fact that more general metrics have to be used for complex data such as microarray data or DNA sequences. Further it might be the case that data are not embedded in a vector space at all, rather, pairwise similarities or dissimilarities are available.

Several extensions of classical SOM and NG to more general data have been proposed: a statistical interpretation of SOM as considered in [5, 14, 34, 36] allows to change the generative model to alternative general data models. The resulting approaches are very flexible but also computationally quite demanding, such that proper

initialization and metaheuristics (e.g. deterministic annealing) become necessary when optimizing statistical models. For specific data structures such as time series or recursive structures, recursive models have been proposed as reviewed e.g. in the article [10]. However, these models are restricted to recursive data structures with euclidean constituents. Online variants of SOM and NG have been extended to general kernels e.g. in the approaches presented in [28, 40] such that the processing of nonlinearly preprocessed data becomes available. However, these versions have been derived for kernels, i.e. similarities and (slow) online adaptation only.

The approach [20] provides a fairly general method for large scale application of SOM to nonvectorial data: it is assumed that pairwise similarities of data points are available. Then the batch optimization scheme of SOM can be generalized by means of the generalized median to a visualization tool for general similarity data. Thereby, prototype locations are restricted to data points. This method has been extended to NG in [3] together with a general proof of the convergence of median versions of clustering. Further developments concern the efficiency of the computation [2] and the integration of prior information if available to achieve meaningful visualization and clustering [6, 7, 37].

Median clustering has the benefit that it builds directly on the derivation of SOM and NG from a cost function. Thus, the resulting algorithms share the simplicity of batch NG and SOM, its mathematical background and convergence, as well as the flexibility to model additional information by means of an extension of the cost function. However, for median versions, prototype locations are restricted to the set of given training data which constitutes a severe restriction in particular for small data sets. Therefore, extensions which allow a smooth adaptation of prototypes have been proposed e.g. in [8]. In this approach, a weighting scheme is introduced for the points which represent virtual prototype locations thus allowing a smooth interpolation between the discrete training data. This model has the drawback that it is not an extension of the standard euclidean version and it gives different results when applied to euclidean data in a real-vector space.

Here, we use an alternative way to extend NG and SOM to relational data given by pairwise similarities or dissimilarities, respectively, which is similar to the relational dual of fuzzy clustering as derived in [12, 13]. For a given euclidean distance matrix or Gram matrix, it is possible to derive the relational dual of topographic map formation which expresses the relevant quantities in terms of the given matrix and which leads to a learning scheme similar to standard batch optimization. This scheme provides identical results as the standard euclidean version if an embedding of the given data points is known. In particular, it possesses the same convergence properties as the standard variants, thereby restricting the computation to known quantities which do not rely on an explicit embedding in the euclidean space. Since these relational variants rely on the same cost function as the standard euclidean batch optimization schemes, extensions to additional label information as proposed for the standard variants [6, 7] become readily available.

In this contribution, we first introduce batch learning algorithms for standard cluster-

ing and topographic map formation derived from a cost function: k-means, neural gas, and the self-organizing map for general (e.g. rectangular, hexagonal, or hyperbolic) grid structures. Then we derive the respective relational dual resulting in a dual cost function and batch optimization schemes for the case of a given distance matrix of data or a given Gram matrix, respectively. We demonstrate the possibility to extend these models to supervised information, and we show the performance in several experiments.

## 2 Topographic maps

Neural clustering and topographic maps constitute effective methods for data preprocessing and visualization. Classical variants deal with vectorial data $\vec{x} \in \mathbb{R}^n$ which are distributed according to an underlying distribution $P$ in the euclidean plane. The goal of neural clustering algorithms is to distribute prototypes $\vec{w}^i \in \mathbb{R}^n$, $i = 1, \ldots, k$ among the data such that they represent the data as accurately as possible. A new data point $\vec{x}$ is assigned to the *winner* $\vec{w}^{I(\vec{x})}$ which is the prototype with smallest distance $\|\vec{w}^{I(\vec{x})} - \vec{x}\|^2$. This clusters the data space into the receptive fields of the prototypes.

Different popular variants of neural clustering have been proposed to learn prototype locations from given training data [19]. Assume the number of prototypes is fixed to $k$. Simple k-means directly optimizes the *quantization error*

$$E_{\mathrm{k-means}}(\vec{w}) = \frac{1}{2} \sum_{i=1}^{k} \int \delta_{i,I(\vec{x})} \cdot \|\vec{x} - \vec{w}^i\|^2 \, P(d\vec{x})$$

where $\delta_{i,I(\vec{x})}$ with Kronecker $\delta$-function indicates the winner neuron for $\vec{x}$. Given a finite set of training data $\vec{x}^1, \ldots, \vec{x}^m$, a batch training algorithm can be directly derived from the cost function, subsequently optimizing the winner assignments, which are treated as hidden variables of the cost function, and the prototype locations:

init $\vec{w}^i$
repeat
  compute optimum assignments $I(\vec{x}^j)$ which minimize $\|\vec{x}^j - \vec{w}^{I(\vec{x}^j)}\|^2$
  compute new prototype locations $\vec{w}^i = \sum_j \delta_{i,I(\vec{x}^j)} \cdot \vec{x}^j / \sum_j \delta_{i,I(\vec{x}^j)}$

K-means constitutes one of the most popular clustering algorithms for vectorial data and can be used as a preprocessing step for data mining and data visualization. However, it is quite sensitive to initialization.

Unlike k-means, neural gas (NG) [23] incorporates the neighborhood of a neuron for adaptation. The cost function is given by

$$E_{\mathrm{NG}}(\vec{w}) = \frac{1}{2C(\lambda)} \sum_{i=1}^{k} \int h_\lambda(k_i(\vec{x})) \cdot \|\vec{x} - \vec{w}^i\|^2 \, P(d\vec{x})$$

where

$$k_i(\vec{x}) = |\{\vec{w}^j \mid \|\vec{x} - \vec{w}^j\|^2 < \|\vec{x} - \vec{w}^i\|^2\}|$$

is the rank of the prototypes sorted according to the distances, $h_\lambda(t) = \exp(-t/\lambda)$ scales the neighborhood cooperation with neighborhood range $\lambda > 0$, and $C(\lambda)$ is the constant $\sum_{i=1}^{k} h_\lambda(k_i(\vec{x}))$. The neighborhood cooperation smoothes the data adaptation such that, on the one hand, sensitivity to initialization can be prevented, on the other hand, a data optimum topological ordering of prototypes is induced by linking the respective two best matching units for a given data point [24]. Classical NG is optimized in an online mode. For a fixed training set, an alternative fast batch optimization scheme is offered by the following algorithm, which in turn computes ranks, which are treated as hidden variables of the cost function, and optimum prototype locations [3]:

init $\vec{w}^i$
repeat
  compute ranks $k_i(\vec{x}^j) = |\{\vec{w}^k \mid \|\vec{x}^j - \vec{w}^k\|^2 < \|\vec{x}^j - \vec{w}^i\|^2\}|$
  compute new prototype locations $\vec{w}^i = \sum_j h_\lambda(k_i(\vec{x}^j)) \cdot \vec{x}^j / \sum_j h_\lambda(k_i(\vec{x}^j))$

Like k-means, NG can be used as a preprocessing step for data mining and visualization, followed e.g. by subsequent projection methods such as multidimensional scaling.

The self-organizing map (SOM) as proposed by Kohonen uses a fixed (usually low-dimensional and regular) lattice structure which determines the neighborhood cooperation. This restriction can induce topological mismatches if the data topology does not match the prior lattice. However, since often a two-dimensional regular lattice is chosen, this has the benefit that, apart from clustering, a direct visualization of the data results by a representation of the data in the regular lattice space. Thus SOM constitutes a direct data inspection and visualization method. SOM itself does not possess a cost function, but a slight variation thereof does, as proposed by Heskes [14]. The cost function is $E_{\mathrm{SOM}}(\vec{w}) =$

$$\frac{1}{2} \sum_{i=1}^{N} \int \delta_{i,I^*(\vec{x})} \cdot \sum_k h_\lambda(n(i,k)) \|\vec{x} - \vec{w}^k\|^2 \, P(d\vec{x})$$

where $n(i,j)$ denotes the neighborhood structure induced by the lattice and $h_\lambda(t) = \exp(-t/\lambda)$ scales the neighborhood degree by a Gaussian function. Thereby, the index $I^*(\vec{x})$ refers to a slightly altered winner notation: the neuron $I^*(\vec{x})$ becomes winner for $\vec{x}$ for which the average distance

$$\sum_k h_\lambda(n(I^*(\vec{x}),k)) \|\vec{x} - \vec{w}^k\|^2$$

is minimum. Often, neurons are arranged in a graph structure which defines the topology, e.g. a rectangular or hexagonal tessellation of the euclidean plane resp. a hyperbolic grid on the two-dimensional hyperbolic plane, the latter allowing a very dense connection of prototypes with exponentially increasing number of neighbors. In these cases, the function $n(i,j)$ denotes the length of a path connecting the prototypes number $i$ and $j$ in the lattice structure. Original SOM is optimized in an online fashion. For fixed

training data, batch optimization is possible by subsequently optimizing assignments and prototype locations:

init
repeat
  compute winner assignments $I^*(\vec{x}^j)$ minimizing $\sum_k h_\lambda(n(I^*(\vec{x}^j), k))\|\vec{x} - \vec{w}^k\|^2$
  compute new prototypes $\vec{w}^i = \sum_j h_\lambda(n(I^*(\vec{x}^j), i) \cdot \vec{x}^j / \sum_j h_\lambda(n(I^*(\vec{x}^j), i)$

It has been shown in e.g. [3] that these batch optimization schemes converge in a finite number of steps towards a (local) optimum of the cost function, provided the data points are not located at borders of receptive fields of the final prototype locations. In the latter case, convergence can still be guaranteed but the final solution can lie at the border of basins of attraction.

# 3 Relational data

Relational data $x^i$ are not contained in a euclidean vector space, rather, pairwise similarities or dissimilarities are available. Batch optimization can be transferred to such situations using the so-called generalized median [3, 20]. Assume, distance information $d(x^i, x^j)$ is available for every pair of data points $x^1, \ldots, x^m$. Median clustering reduces prototype locations to data locations, i.e. adaptation of prototypes is not continuous but takes place within the space $\{x^1, \ldots, x^m\}$ given by the data. We write $w^i$ to indicate that the prototypes need no longer be vectorial. For this restriction, the same cost functions as beforehand can be defined whereby the euclidean distance $\|\vec{x}^j - \vec{w}^i\|^2$ is substituted by $d(x^j, w^i) = d(x^j, x^{l_i})$ whereby $w^i = x^{l_i}$. Median clustering substitutes the assignment of $\vec{w}^i$ as (weighted) center of gravity of data points by an extensive search, setting $w^i$ to the data points which optimize the respective cost function for fixed assignments. This procedure has been tested e.g. in [3, 6]. It has the drawback that prototypes have only few degrees of freedom if the training set is small. Thus, median clustering usually gives inferior results compared to the classical euclidean versions when applied in a euclidean setting.

Here we introduce relational clustering for data characterized by similarities or dissimilarities, whereby this setting constitutes a direct transfer of the standard euclidean training algorithm to more general settings allowing smooth updates of the solutions. The essential observation consists in a transformation of the cost functions as defined above to their so-called relational dual. We distinguish two settings, similarity data where dot products of training data are available, and dissimilarity data where pairwise distances are available.

## 3.1 Metric data

Assume training data $x^1, \ldots, x^m$ are given in terms of pairwise distances $d_{ij} = d(x^i, x^j)^2$. We assume that it originates from a euclidean distance measure, that means,

the triangle inequality holds for $d$ and we are always able to find (possibly high dimensional) euclidean points $\vec{x}^i$ such that $d_{ij} = \|\vec{x}^i - \vec{x}^j\|^2$. Note that this notation includes a possibly nonlinear mapping (feature map) $x^i \mapsto \vec{x}^i$ corresponding to the embedding in a euclidean space. However, this embedding is not known, such that we cannot directly optimize the above cost functions in the embedding space.

Median clustering restricts prototype locations to given data points and determines the prototype locations in each iterative step in such a way that the corresponding part of the cost function (assumed fixed assignments) becomes minimum. These values are determined by extensive search, turning the linear complexity to a quadratic one w.r.t. the number of training data. This procedure has the severe drawback that only discrete adaptation steps can be performed and the result is usually worse compared to standard SOM or NG in the euclidean setting.

Relational learning overcomes this problem. The key observation consists in the fact that optimum prototype locations $\vec{w}^j$ can be expressed as linear combination of data points. Therefore, the unknown distances $\|\vec{x}^j - \vec{w}^i\|^2$ can be expressed in terms of known values $d_{ij}$.

More precisely, assume there exist points $\vec{x}^j$ such that $d_{ij} = \|\vec{x}^i - \vec{x}^j\|^2$. Assume the prototypes can be expressed in terms of data points $\vec{w}^i = \sum_j \alpha_{ij}\vec{x}^j$ where $\sum_j \alpha_{ij} = 1$. Then $\|\vec{w}^i - \vec{x}^j\|^2 = (D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$ where $D = (d_{ij})_{ij}$ constitutes the distance matrix and $\alpha_i = (\alpha_{ij})_j$ the coefficients.

This fact can be shown as follows: for $\vec{w}^i = \sum_j \alpha_{ij}\vec{x}^j$, one can compute $\|\vec{x}^j - \vec{w}^i\|^2 = \|\vec{x}^j\|^2 - 2\sum_l \alpha_{il}(\vec{x}^j)^t\vec{x}^l + \sum_{l,l'} \alpha_{il}\alpha_{il'}(\vec{x}^l)^t\vec{x}^{l'}$ which is the same as $(D\cdot\alpha_i)_j - 1/2\cdot\alpha_i^t\cdot D\cdot\alpha_i = \sum_l \|\vec{x}^j - \vec{x}^l\|^2\cdot\alpha_{il} - 1/2\cdot\sum_{ll'}\alpha_{il}\|\vec{x}^l - \vec{x}^{l'}\|^2\alpha_{il'} = \sum_l \|\vec{x}^j\|^2\alpha_{il} - 2\cdot\sum_l \alpha_{il}(\vec{x}^j)^t\vec{x}^l + \sum_l \alpha_{il}\|(\vec{x}^l)\|^2 - \sum_{ll'}\alpha_{il'}\alpha_{il'}\|\vec{x}^l\|^2 + \sum_{ll'}\alpha_{il}\alpha_{il'}(\vec{x}^l)^t\vec{x}^{l'}$ because of $\sum_j \alpha_{ij} = 1$.

Because of this fact, we can substitute all terms $\|\vec{x}^j - \vec{w}^i\|^2$ in batch optimization schemes. For optimum solutions we find the equality $\vec{w}^i = \sum_j \alpha_{ij}\vec{x}^j$ for all three batch optimization schemes as introduced above, whereby

1. $\alpha_{ij} = \delta_{i,I(\vec{x}^j)} / \sum_j \delta_{i,I(\vec{x}^j)}$ for k-means,

2. $\alpha_{ij} = h_\lambda(k_i(\vec{x}^j)) / \sum_j h_\lambda(k_i(\vec{x}^j))$ for NG, and

3. $\alpha_{ij} = h_\lambda(n(I^*(\vec{x}^j), i)) / \sum_j h_\lambda(n(I^*(\vec{x}^j), i)$ for SOM.

This allows to reformulate the batch optimization schemes in terms of relational data.

We obtain the algorithm

init $\alpha_{ij}$ with $\sum_j \alpha_{ij} = 1$
repeat
  compute $\|\vec{x}^j - \vec{w}^i\|^2$ as $(D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$
  compute optimum assignments based on this distance matrix
    $\tilde{\alpha}_{ij} = \delta_{i,I(\vec{x}^j)}$ (for k-means)
    $\tilde{\alpha}_{ij} = h_\lambda(k_i(\vec{x}^j))$ (for NG)
    $\tilde{\alpha}_{ij} = h_\lambda(n(I^*(\vec{x}^j), i))$ (for SOM)
  compute $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$ as normalization of these values.

Hence, prototype locations are computed only indirectly by means of the coefficients $\alpha_{ij}$. Initialization can be done e.g. setting initial prototype locations to random data points, which is realized by a random selection of $N$ rows from the given distance matrix.

Given a new data point $x$ which can isometrically be embedded in euclidean space as $\vec{x}$, and pairwise distances $d_j = d(x, x^j)^2$ corresponding to the distance from $x^j$, the winner can be determined by using the equality

$$\|\vec{x} - \vec{w}^i\|^2 = (D(x)^t \cdot \alpha_i) - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$$

where $D(x)$ denotes the vector of distances $D(x) = (d_j)_j = (d(x, x^j)^2)_j$. This holds because $\|\vec{x} - \vec{w}^i\|^2 = \|\vec{x}\|^2 - 2\sum_l \alpha_{il} \vec{x}^t \vec{x}^j + \sum_{ll'} \alpha_{il}\alpha_{il'} (\vec{x}^l)^t \vec{x}^{l'}$ which is the same as the latter term, which equals $\sum_l \alpha_{il} \|\vec{x} - \vec{x}^l\|^2 - 1/2 \sum_{ll'} \alpha_{il}\alpha_{il'} \|\vec{x}^l - \vec{x}^{l'}\|^2$, because of $\sum_l \alpha_{il} = 1$.

The quantization error can be expressed in terms of the given values $d_{ij}$ by substituting $\|\vec{x}^j - \vec{w}^i\|^2$ by $(D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$. Using the formula for optimum assignments of batch optimization, one can also derive relational dual cost functions for the algorithms. For k-means, we use the shorthand notation $\delta_{ij} = \delta_{i,I(\vec{x}^j)}$. It holds $\vec{w}^i = \sum_j \delta_{ij} \cdot \vec{x}^j / \sum_j \delta_{ij}$, hence the cost function becomes $1/2 \cdot \sum_{ij} \delta_{ij} \|\vec{w}^i - \vec{x}^j\|^2 = 1/2 \cdot \sum_{ij} \delta_{ij} \| \sum_l \delta_{il} \vec{x}^l / \sum_l \delta_{il} \|^2 = \sum_i 1/(2 \sum_l \delta_{il}) \cdot \left( \sum_{ll'} \delta_{il}\delta_{il'} \|\vec{x}^l\|^2 - \sum_{ll'} \delta_{il}\delta_{il'} (\vec{x}^l)^t \vec{x}^{l'} \right)$. Thus, the relational dual of k-means is

$$\sum_i \frac{1}{4 \cdot \sum_l \delta_{iI(\vec{x}^l)}} \cdot \sum_{ll'} \delta_{iI(\vec{x}^l)} \delta_{iI(\vec{x}^{l'})} d_{ll'} \, .$$

This measures the pairwise distance of data points assigned to the same cluster.

For NG, we use the abbreviation $k_{ij} = h_\lambda(k_i(\vec{x}^j))$. Because of $\vec{w}^i = \sum_j k_{ij} \cdot \vec{x}^j / \sum_j k_{ij}$, we find $1/2 \cdot \sum_{ij} k_{ij} \|\vec{x}^j - \vec{w}^i\|^2 = 1/2 \cdot \sum_{ij} k_{ij} \|\vec{x}^j - \sum_l k_{il} \cdot \vec{x}^l / \sum_l k_{il}\|^2 = \sum_i 1/(2 \cdot \sum_l k_{il}) \cdot \left( \sum_{ll'} k_{il}k_{il'} \|\vec{x}^l\|^2 - \sum_{ll'} k_{il}k_{il'} (\vec{x}^l)^t \vec{x}^{l'} \right)$. Thus, the relational dual of NG is

$$\sum_i \frac{1}{4 \sum_l h_\lambda(k_i(\vec{x}^l))} \cdot \sum_{ll'} h_\lambda(k_i(\vec{x}^l)) h_\lambda(k_i(\vec{x}^{l'})) d_{ll'} \, .$$

Obviously, this extends the relational dual of k-means towards neighborhood cooperation.

For SOM, we can rewrite the cost function as

$$\frac{1}{2}\sum_{ij} h_\lambda(n(I^*(\vec{x}^j),i)\|\vec{x}^j - \vec{w}^i\|^2$$

for the discrete setting. We use the shorthand notation $n_{ij} = h_\lambda(n(I^*(\vec{x}^j),i)$. Because of $\vec{w}^i = \sum_j n_{ij}\cdot\vec{x}^j / \sum_j n_{ij}$ we find $1/2\cdot\sum_{ij} n_{ij}\|\vec{x}^j - \vec{w}^i\|^2 = 1/2\cdot\sum_{ij} n_{ij}\|\vec{x}^j - \sum_l n_{il}\cdot\vec{x}^l / \sum_l n_{il}\|^2 = \sum_i 1/(2\sum_l n_{il})\cdot\left(\sum_{ll'} n_{il}n_{il'}\|\vec{x}^l\|^2 - \sum_{ll'} n_{il}n_{il'}(\vec{x}^l)^t\vec{x}^{l'}\right)$ hence the relational dual is

$$\sum_i \frac{1}{4\sum_l h_\lambda n(I^*(\vec{x}^l),i)} h_\lambda(n(I^*(\vec{x}^l),i))h_\lambda(n(I^*(\vec{x}^{l'}),i))d_{ll'}$$

thus extending k-means to neighborhood cooperation as induced by the lattice.

Note that this relational learning gives exactly the same results as standard batch optimization provided the given relations stem from an euclidean metric. Hence, convergence is guaranteed in this case since it holds for the standard batch versions. If the given distance matrix does not stem from an euclidean metric, this equality does no longer hold and the terms $(D\cdot\alpha_i)_j - 1/2\cdot\alpha_i^t\cdot D\cdot\alpha_i$ can become negative. In this case, one can correct the distance matrix by the $\gamma$-spread transform $D_\gamma = D + \gamma(\mathbf{1} - \mathbf{I})$ for sufficiently large $\gamma$ where $\mathbf{1}$ equals 1 for each entry and $\mathbf{I}$ is the identity.

## 3.2 Dot products

A dual possibility is to characterize data $x^1, \ldots, x^m$ by means of pairwise similarities, i.e. dot products. We denote the similarity of $x^i$ and $x^j$ by $k(x^i, x^j) = k_{ij}$. We assume that these values fulfill the properties of a dot product, i.e. the matrix $K$ with entries $k_{ij}$ is positive definite. In this case, a representation $\vec{x}^i$ of the data can be found in a possibly high dimensional euclidean vector space such that $k_{ij} = (\vec{x}^i)^t\vec{x}^j$.

As beforehand, we can represent distances in terms of these values if $\vec{w}^i = \sum_l \alpha_{il}\vec{x}^l$ with $\sum_l \alpha_{il} = 1$ yields optimum prototypes:

$$\|\vec{x}^j - \vec{w}^i\|^2 = k_{jj} - 2\sum_l \alpha_{il}k_{jl} + \sum_{ll'}\alpha_{il}\alpha_{il'}k_{ll'}\ .$$

This allows to compute batch optimization in the same way as beforehand:

init $\alpha_{ij}$ with $\sum_j \alpha_{ij} = 1$
repeat
  compute the distance $\|\vec{x}^j - \vec{w}^i\|^2$ as $k_{jj} - 2\sum_l \alpha_{il}k_{jl} + \sum_{ll'}\alpha_{il}\alpha_{il'}k_{ll'}$
  compute optimum assignments based on this distance matrix
    $\tilde{\alpha}_{ij} = \delta_{i,I(\vec{x}^j)}$ (for k-means)
    $\tilde{\alpha}_{ij} = h_\lambda(k_i(\vec{x}^j))$ (for NG)
    $\tilde{\alpha}_{ij} = h_\lambda(n(I^*(\vec{x}^j),i))$ (for SOM)
  compute $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$ as normalization of these values.

One can use the same identity for $\|\vec{x} - \vec{w}^i\|^2$ to obtain a possibility to compute the winner given a point $x$ and to compute the respective cost function. Convergence of this algorithm is guaranteed since it is identical to the batch versions for the euclidean data embedding $\vec{x}^i$ if $K$ is positive definite.

If $K$ is not positive definite negative values can occur for $\|\vec{x}^j - \vec{w}^i\|^2$. In this case, the kernel matrix can be corrected by the operation $K_\gamma = K + \gamma \cdot \mathbf{1}$ with large enough $\gamma$.

# 4 Supervision

The possibility to include further information, if available, is very important to get meaningful results for unsupervised learning. This can help to prevent the 'garbage in - garbage out' problem of unsupervised learning, as discussed e.g. in [17, 18]. Here we assume that additional label information is available which should be accounted for by clustering or visualization. Thereby, labels are embedded in $\mathbb{R}^d$ and can be fuzzy. We assume that the label attached to $x^j$ is denoted by $\vec{y}^j$. We equip a prototype $w^i$ with a label $\vec{Y}^i \in \mathbb{R}^d$ which is adapted during learning. For the euclidean case, the basic idea consists in a substitution of the standard euclidean distance $\|\vec{x}^j - \vec{w}^i\|^2$ by a mixture

$$(1-\beta) \cdot \|\vec{x}^j - \vec{w}^i\|^2 + \beta \cdot \|\vec{y}^j - \vec{Y}^i\|^2$$

which takes the similarity of label assignments into account and where $\beta \in [0,1]$ controls the influence of the label values. This procedure has been proposed in [6, 7, 37] for euclidean and median clustering and online neural gas, respectively. One can use the same principles to extend relational clustering.

For discrete euclidean settings $\vec{x}^1, \ldots, \vec{x}^m$ cost functions and related batch optimization is as follows (neglecting constant factors):

$$E_{\mathrm{k-means}}(\vec{w}, \vec{Y}) = \sum_{ij} \delta_{i,I(\vec{x}^j)} \cdot \left( (1-\beta) \cdot \|\vec{x}^j - \vec{w}^i\|^2 + \beta \cdot \|\vec{y}^j - \vec{Y}^i\|^2 \right)$$

where $\delta_{i,I(\vec{x}^j)}$ indicates the winner for $\vec{x}^j$ which is the neuron $\vec{w}^{I(\vec{x}^j)}$ with smallest $(1-\beta) \cdot \|\vec{x}^j - \vec{w}^{I(\vec{x}^j)}\|^2 + \beta \cdot \|\vec{y}^j - \vec{Y}^{I(\vec{x}^j)}\|^2$. Besides this slight change in the winner notation, the batch update is extended by the adaptation step $\vec{Y}^i = \sum_j \delta_{i,I(\vec{x}^j)} \vec{y}^j / \sum_j \delta_{i,I(\vec{x}^j)}$ for the prototype labels.

Similarly, the cost function of NG becomes

$$E_{\mathrm{NG}}(\vec{w}, \vec{Y}) = \sum_{ij} h_\lambda(k_i(\vec{x}^j)) \cdot \left( (1-\beta) \cdot \|\vec{x}^j - \vec{w}^i\|^2 + \beta \cdot \|\vec{y}^j - \vec{Y}^i\|^2 \right)$$

where $k_i(\vec{x}^j)$ denotes the rank of neuron $i$ measured according to the distances $(1-\beta) \cdot \|\vec{x}^j - \vec{w}^i\|^2 + \beta \cdot \|\vec{y}^j - \vec{Y}^i\|^2$. Again, this change in the computation of the rank

is accompanied by the adaptation $\vec{Y}^i = \sum_j h_\lambda(\vec{x}^j)\vec{y}^j / \sum_j h_\lambda(\vec{x}^j)$ for the prototype labels for batch optimization

In the same way, the cost function of SOM becomes

$$E_{\mathrm{SOM}}(\vec{w}, \vec{Y}) = \delta_{i,I^*(\vec{x}^j)} \sum_k h_\lambda(n(i,k)) \\ \Big((1-\beta)\cdot\|\vec{x}^j - \vec{w}^k\|^2 + \beta\cdot\|\vec{y}^j - \vec{Y}^k\|^2\Big)$$

where $I^*(\vec{x}^j)$ denotes the generalization of the winner notation proposed by Heskes to the supervised setting, i.e. it is the prototype which minimizes $\sum_k h_\lambda(n(I^*\vec{x}^j), k))$ $\Big((1-\beta)\cdot\|\vec{x}^j - \vec{w}^k\|^2 + \beta\cdot\|\vec{y}^j - \vec{Y}^k\|^2\Big)$. Batch optimization uses this winner notation and extends the updates by $\vec{Y}^i = \sum_j h_\lambda(n(i, I^*(\vec{x}^j))\vec{y}^j / \sum_j h_\lambda(n(i, I^*(\vec{x}^j))$. It has been shown in [6, 7] that these procedures converge in a finite number of steps in the same way as the original unsupervised versions.

For these generalized cost functions, relational learning becomes possible by substituting the distances $\|\vec{x}^j - \vec{w}^i\|^2$ using the identity $\vec{w}^i = \sum \alpha_{ij}\vec{x}^j$ for optimum assignments which still holds for these extensions. The same computation as beforehand yields to the algorithm for clustering dissimilarity data characterized by pairwise distances $d_{ij}$:

init $\alpha_{ij}$ with $\sum_j \alpha_{ij} = 1$
repeat
  compute the distances as $(1-\beta)\cdot((D\cdot\alpha_i)_j - 1/2\cdot\alpha_i^t D\alpha_i) + \beta\cdot\|Y^i - y^j\|^2$
  compute optimum assignments $\tilde{\alpha}_{ij}$ based on this distance as before
  compute $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$
  compute prototype labels $\vec{Y}^i = \sum_j \alpha_{ij}\vec{y}^j$

An extension to similarity data given by dot products $\vec{x}^i \cdot \vec{x}^j$ proceeds in the same way using the different distance computation based on dot products as derived beforehand.

Since this version is identical to the euclidean version for distance data resp. dot products, this procedure converges in a finite number of steps to a solution. Unlike a pure unsupervised learning with posterior labeling, the prototypes are forced to follow the borders given by the class information for the supervised setting. If the triangle inequality or positive definiteness do not hold negative distances can occur, in which case a correction of the given matrix is advisable. Note that, for vanishing neighborhood size of NG and SOM, the final prototype labels coincide with the averaged label taken over the receptive field of a prototype. For rapid learning, one can improve the classification result by setting the prototype labels to the averaged label of the receptive fields after training. Note that, still, the prototype locations are affected by the label information, unlike a pure unsupervised learning with posterior labeling. For the supervised setting, the prototypes are forced to follow the borders given by the class information.

# 5 Visualization

As discussed before, data can be clustered using trained prototypes. But it is not obvious whether we are able to create a visualization of such extracted information. For low-dimensional euclidean SOM, a direct visualization is given by an embedding of the data points to the respective positions of their winner in the lattice space. For the computation of these distances, the formulas as derived above can be applied. Thereby, the euclidean coordinates of neurons of a rectangular SOM are directly given by their respective index in the lattice: neuron number $i$ in row $j$ is located at position $(i, j)$ or a scaled variant thereof. For a hyperbolic lattice structure, the Poincaré disk model of the hyperbolic plane can be used [1] which embeds the hyperbolic space non-isometrically into the unit disk such that the focus of attention is put onto the point which is mapped into the center of the disk and an overall fish-eye effect results. Moving this focus allows to browse through the map.

For NG and k-means, no direct visualization is given, but it can be easily reached by a subsequent embedding of the prototypes into the two-dimensional euclidean plane by means of distance preserving projection techniques of the prototypes such as multidimensional scaling. Given pairwise distances $\delta_{ij} := \|\vec{w}^i - \vec{w}^j\|$ of the prototypes (possibly nonlinearly preprocessed, i.e. $\delta_{ij} = f(\|\vec{w}^i - \vec{w}^j\|)$ where $f$ is an appropriate weighting function), this model finds two-dimensional projections $p(\vec{w}^i)$ with pairwise distances $\Delta_{ij} := \|p(\vec{w}^i) - p(\vec{w}^j)\|$ such that the stress-function

$$\left( \frac{\sum_{i<j} (\delta_{ij} - \Delta_{ij})^2}{\sum_{i<j} \Delta_{ij}^2} \right)^2$$

or a similar objective function is minimized.

In order to apply these techniques, the pairwise distance of prototypes needs to be computed. As beforehand, we assume the identity $\vec{w}^i = \sum_l \alpha_{il} \vec{x}^l$ for optimum prototypes $\vec{w}^i$. Then one can compute $\|\vec{w}^i - \vec{w}^j\|^2 = \|\sum_l \alpha_{il}\vec{x}^l - \sum_{l'} \alpha_{jl'}\vec{x}^{l'}\|^2 = \sum_{ll'} (\alpha_{il}\alpha_{il'} + \alpha_{jl}\alpha_{jl'} - 2\alpha_{il}\alpha_{jl'}) (\vec{x}^l)^t \vec{x}^{l'}$ which can be directly computed assumed pairwise dot products $k_{ij} = (\vec{x}^l)^t \vec{x}^{l'}$ of the data are available.

If pairwise distances $d_{ij} = \|\vec{x}^i - \vec{x}^j\|^2$ are given, accumulated in a distance matrix $D$ as beforehand, we find the identity

$$\|\vec{w}^i - \vec{w}^j\|^2 = \alpha_j^t D \alpha_i - \frac{1}{2}\alpha_j^t D \alpha_j - \frac{1}{2}\alpha_i^t D \alpha_i$$

where $\alpha_i$ denotes the vector $(\alpha_{il})_l$. This holds because the term yields $\sum_{ll'} \alpha_{jl}\alpha_{il'} \|\vec{x}^l - \vec{x}^{l'}\|^2 - 1/2 \cdot \sum_{ll'} \alpha_{jl}\alpha_{jl'} \|\vec{x}^l - \vec{x}^{l'}\|^2 - 1/2 \cdot \sum_{ll'} \alpha_{il}\alpha_{il'} \|\vec{x}^l - \vec{x}^{l'}\|^2 = \sum_l \alpha_{jl} \|\vec{x}^l\|^2 - 2 \cdot \sum_{ll'} \alpha_{il}\alpha_{jl'} (\vec{x}^l)^t \vec{x}^{l'} + \sum_l \alpha_{il} \|\vec{x}^l\|^2 - \sum_l \alpha_{jl} \|\vec{x}^l\|^2 + \sum_{ll'} \alpha_{jl}\alpha_{jl'} (\vec{x}^l)^t \vec{x}^{l'} - \sum_l \alpha_{il} \|\vec{x}^l\|^2 + \sum_{ll'} \alpha_{il}\alpha_{il'} (\vec{x}^l)^t \vec{x}^{l'}$ which is the same as the above term. Thus, pairwise distances of prototypes can be computed in terms of the given distance matrix $D$ of the data.

# 6 Experiments

In the experiments we demonstrate the performance of the neural gas and k-means algorithms in different scenarios covering several characteristic situations. All algorithms have been implemented based on the SOM Toolbox for Matlab [27]. Note that, for all median versions, prototypes situated at identical points of the data space do not separate in subsequent runs. Therefore constellations with exactly identical prototypes should be avoided. For the euclidean and relational versions this problem is negligible, presumed prototypes are initialized at different positions. However, for median versions it is likely that prototypes move to an identical locations due to the limited number of different positions in data space, in particular for small data sets. To cope with this fact in median versions, we add a small amount of noise to the distances in each epoch in order to separate identical prototypes. The initial neighborhood rate for neural gas is $\lambda = n/2$, $n$ being the number of neurons, and it is multiplicatively decreased during training.

## Wisconsin Breast Cancer Database

The Wisconsin Diagnostic Breast Cancer database (WDBC) is a standard benchmark set from clinical proteomics [39]. It consists of 569 data points described by 30 real-valued input features: digitized images of a fine needle aspirate of breast mass are described by characteristics such as form and texture of the cell nuclei present in the image. Data are labeled by two classes, benign and malignant.

For training we used 40 neurons and 150 epochs per run. The dataset was z-transformed beforehand. The results were gained from repeated 2-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 1. Moreover, the data set is contained in the Euclidean space therefore we are able to compare the relational versions introduced in this article to the standard euclidean methods. These results are shown in Table 1. The effect of a variation of the mixing parameter is demonstrated in Fig. **??**. The results are competitive to supervised learning with the state-of-the-art-method GRLVQ as obtained in the approach [33].

As one can see, the results of euclidean and relational clustering are identical, as expected by the theoretical background of relational clustering. Relational clustering and supervision allow to improve the more restricted and unsupervised median versions by more than 1% classification accuracy.

## Cat Cortex

The Cat Cortex Data Set originates from anatomic studies of cats' brains. A matrix of connection strengths between 65 cortical areas of cats was compiled from literature [4]. There are four classes corresponding to four different regions of the cortex. For our

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch NG | Supervised Relational k-Means | Supervised Relational Batch NG |
|---|---|---|---|---|---|---|---|
| Accuracy | | | | | | | |
| Mean | 93.0 | 93.1 | 93.4 | 94.0 | 94.3 | 93.5 | 94.4 |
| StdDev | 1.0 | 1.0 | 1.2 | 0.9 | 0.9 | 1.1 | 1.0 |
| | k-Means | Supervised k-Means | Batch NG | Supervised Batch NG | Relational Batch NG | Supervised Relational Batch NG | |
| Accuracy | | | | | | | |
| Mean | 93.6 | 93.0 | 94.1 | 94.7 | 94.0 | 94.3 | |
| StdDev | 0.8 | 1.1 | 1.0 | 0.8 | 0.9 | 0.9 | |

Table 1: Classification accuracy on the Wisconsin Diagnostic Breast Cancer database for posterior labeling. The mean accuracy over 100 repeats of 2-fold cross-validation is reported.

experiments a preprocessed version of the data set from Haasdonk et al. [11] was used. The matrix is symmetric but the triangle inequality does not hold.

The algorithms were tested in 10-fold cross-validation using 12 neurons (three per class) and 150 epochs per run. The results presented reveal the mean accuracy over 250 repeated 10-fold cross-validations per method. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 2. Results for different mixing parameters are shown in Figure 2.

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch NG | Supervised Relational k-Means | Supervised Relational Batch NG |
|---|---|---|---|---|---|---|---|
| Accuracy | | | | | | | |
| Mean | 72.8 | 71.6 | 89.0 | 88.7 | 77.9 | 89.2 | 91.3 |
| StdDev | 3.9 | 4.0 | 3.3 | 3.0 | 3.5 | 3.0 | 2.8 |

Table 2: Classification accuracy on the Cat Cortex Data Set for posterior labelling. The mean accuracy over 250 repeats of 10-fold cross-validation is reported.

A direct comparison of our results to the findings of Graepel et al. [4] or Haasdonk et al. [11] is not possible. Haasdonk et al. gained an accumulated error over all classes of at least 10% in leave-one-out experiments with SVMs. Graepel et al. obtained virtually the same results with the Optimal Hyperplane (OHC) algorithm. In our experiments, the improvement of restricted median clustering by relational extensions can clearly be observed, which accounts for more than 10% classification accuracy. Note that relational clustering works quite well, in this case although a theoretical foundation due to the non-metric similarity matrix is missing.

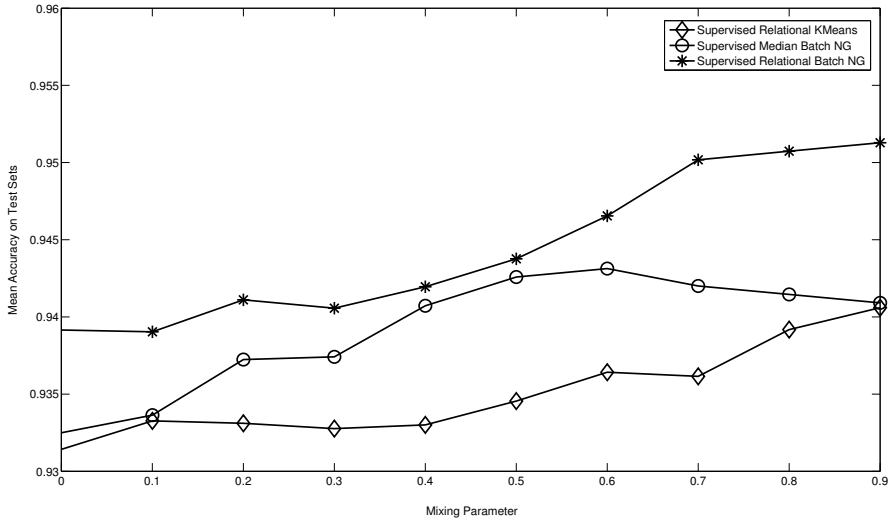


Figure 1: Results of the supervised methods for the Wisconsin Diagnostic Breast Cancer database with different mixing parameters applied.

## Proteins

The evolutionary distance of 226 globin proteins is determined by alignment as described in [25]. These samples originate from different protein families: hemoglobin-$\alpha$, hemoglobin-$\beta$, myoglobin, etc. Here, we distinguish five classes as proposed in [11]: HA, HB, MY, GG/GP, and others. Table 3 shows the class distribution of the dataset.

| Class No. | Count | Percentage |
|---|---|---|
| HA | 72 | 31.86% |
| HB | 72 | 31.86% |
| MY | 39 | 17.26% |
| GG/GP | 30 | 13.27% |
| Others | 13 | 5.75% |

Table 3: Class Statistics of the Protein Dataset

For training we used 45 neurons and 150 epochs per run. The results were gained from repeated 10-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 4. Figure 3 shows results for varying mixing parameters.

Unlike the results reported in [11] for SVM which uses one-versus-rest encoding, the classification in our setting is given by only one clustering model. Depending on

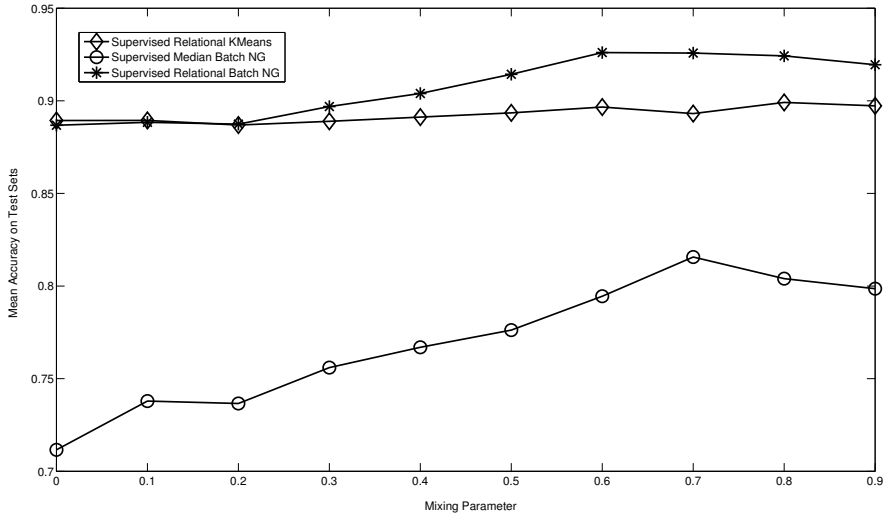


Figure 2: Results of the supervised methods for the Cat Cortex Data Set with different mixing parameters applied.

the choice of the kernel, [11] reports errors which approximately add up to 4% for the leave-one-out error. This result, however, is not comparable to our results due to the different error measure. A 1-nearest neighbor classifier yields an accuracy 91.6 for our setting (k-nearest neighbor for larger k is worse; [11] which is comparable to our results. Thereby, continuous updates improve the results found by median clustering by 14%.

The projections of a Relational SOM with hyperbolic grid structure and of Relational BNG with non-metric multidimensional scaling using Kruskal's normalized stress1 criterion are shown in figures 4 and 5. The neurons are depicted according to majority

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch NG | Supervised Relational k-Means | Supervised Relational Batch NG |
|---|---|---|---|---|---|---|---|
| Accuracy | | | | | | | |
| Mean | 76.1 | 76.3 | 88.0 | 89.9 | 89.4 | 88.2 | 90.0 |
| StdDev | 1.3 | 1.8 | 1.8 | 1.3 | 1.4 | 1.7 | 1.0 |

Table 4: Classification accuracy on the Protein Data Set for posterior labeling. The mean accuracy over 100 repeats of 10-fold cross-validation is reported.

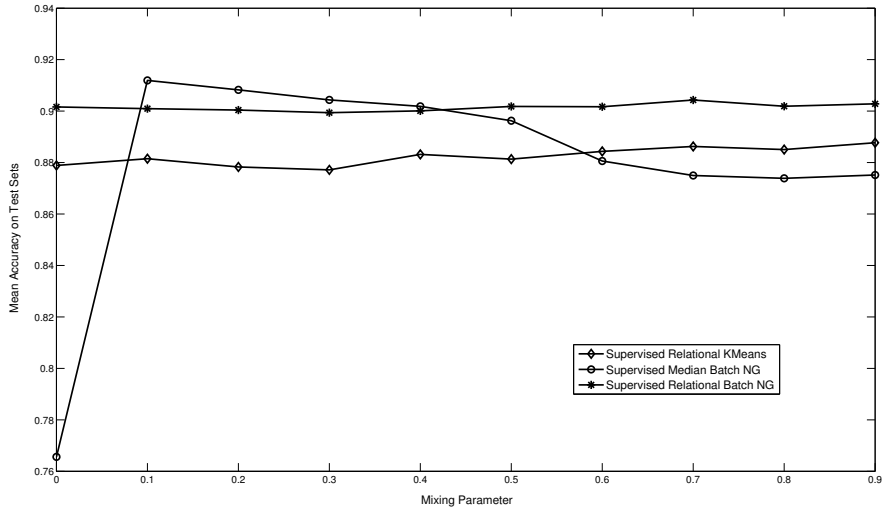


Figure 3: Results of the supervised methods for the Protein Data Set with different mixing parameters applied.

vote. Obviously, the neurons arrange according to the associated class and a very clear two-dimensional representation of the data set is obtained this way.

## Chromosomes

The Copenhagen chromosomes database is a benchmark from cytogenetics [22]. A set of 4200 human nuclear chromosomes from 22 classes (the X resp. Y sex chromosome is not considered) are represented by the grey levels of their images and transferred to strings representing the profile of the chromosome by the thickness of their silhouettes. Thus, this data set is non-Euclidean, consisting of strings of different length, and standard k-means clustering cannot be used. Median versions, however, are directly applicable. The edit distance is a typical distance measure for two strings of different length, as described in [16, 26]. In our application, distances of two strings are computed using the standard edit distance whereby substitution costs are given by the signed difference of the entries and insertion/deletion costs are given by 4.5 [26].

The algorithms were tested in 2-fold cross-validation using 100 neurons and 100 epochs per run (cf. [3]). The results presented are the mean accuracy over 10 times 2-fold cross-validation per method. The mixing parameter of the supervised methods was set to 0.9.

As can be seen, supervised relational neural gas achieves an accuracy of 0.914 for

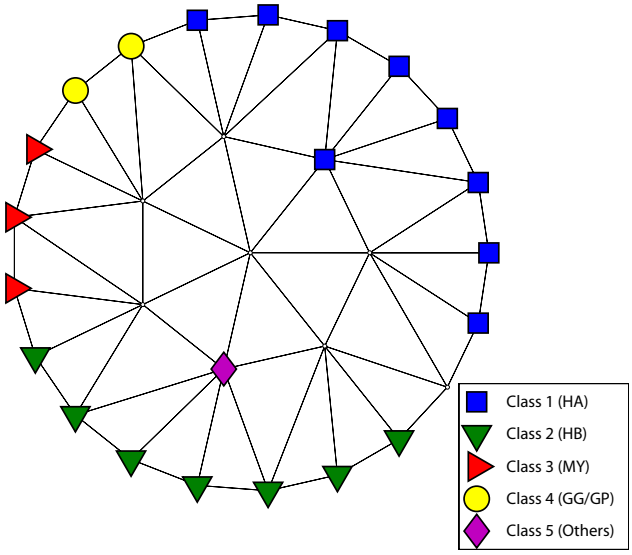


Figure 4: Mapping of the non-euclidean Protein dataset by a Relational SOM with hyperbolic grid structure.

$\alpha = 0.9$. This improves by 8% compared to median variants.

# 7 Discussion

We have introduced relational extensions of neural clustering which extend the classical euclidean versions to settings where pairwise distances or dot products of the data are given but no explicit embedding into a euclidean space is known. By means of the relational dual, batch optimization can be formulated in terms of these quantities only. This extends previous median clustering variants to a continuous prototype update which is particularly useful for only sparsely sampled data. The derived relational algorithms

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch NG | Supervised Relational k-Means | Supervised Relational Batch NG |
|---|---|---|---|---|---|---|---|
| Accuracy | | | | | | | |
| Mean | 82.3 | 82.8 | 90.6 | 91.3 | 89.4 | 90.1 | 91.4 |
| StdDev | 2.2 | 1.7 | 0.6 | 0.2 | 0.6 | 0.6 | 0.6 |

Table 5: Classification accuracy on the Copenhagen Chromosome Database for posterior labeling. The mean accuracy over 10 runs of 2-fold cross-validation is reported.

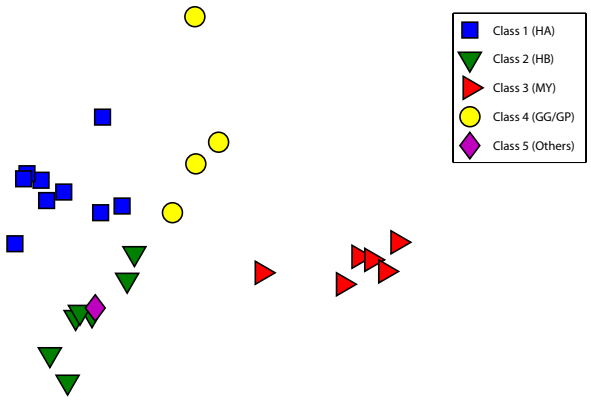


Figure 5: Mapping of the non-euclidean Protein dataset by Relational BNG with non-metric multidimensional scaling.

have a formal background only for euclidean distances or metrics; however, as demonstrated in an example for the cat cortex data, the algorithms might also prove useful in more general scenarios.

The general framework as introduced in this article opens the way towards the transfer of further principles of SOM and NG to the setting of relational data: as an example, the magnification factor of topographic map formation for relational data transfers from the euclidean space, and possibilities to control this factor as demonstrated for batch clustering e.g. in the approach [9] can readily be used.

One very important subject of future work concerns the complexity of computation and sparseness of prototype representation. For the approach as introduced above, the complexity scales quadratic with the number of training examples and For the approach as introduced above, the complexity scales quadratic with the number of training examples and the size of prototype representations is linear with respect to the number of examples. For SOM, it would be worthwhile to investigate whether efficient alternative computation schemes such as proposed in the approach [2] can be derived. Further, the representation contains a large number of very small coefficients, which correspond to data points for which the distance from the prototype is large. Therefore it can be expected that a restriction of the representation to the close neighborhood is sufficient for accurate results.

# References

[1] James W. Anderson (2005), Hyperbolic Geometry, second edition, Springer.

[2] B. Conan-Guez, F. Rossi, and A. El Golli (2006), Fast Algorithm and Implementation of Dissimilarity Self-Organizing Maps, *Neural Networks* 19:855-863.

[3] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann (2006), Batch and median neural gas, *Neural Networks* 19:762-771.

[4] T. Graepel, R. Herbrich, P. Bollmann-Sdorra and K. Obermayer (1999), Classification on pairwise proximity data, In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), NIPS, vol. 11, MIT Press, p. 438-444.

[5] T. Graepel and K. Obermayer (1999), A stochastic self-organizing map for proximity data, *Neural Computation* 11:139-155.

[6] B. Hammer, A. Hasenfuss, F.-M. Schleif, and T. Villmann (2006), Supervised median neural gas, In Dagli, C., Buczak, A., Enke, D., Embrechts, A., and Ersoy, O. (Eds.), *Intelligent Engineering Systems Through Artificial Neural Networks* 16, Smart Engineering System Design, pp.623-633, ASME Press.

[7] B. Hammer, A. Hasenfuss, F.-M. Schleif, and T. Villmann (2006), Supervised batch neural gas, In *Proceedings of Conference Artificial Neural Networks in Pattern Recognition (ANNPR)*, F. Schwenker (ed.), Springer, pages 33-45.

[8] A. Hasenfuss, B. Hammer, F.-M. Schleif, and T. Villmann (2007), Neural gas clustering for dissimilarity data with continuous prototypes, accepted for IWANN'07.

[9] B. Hammer, A. Hasenfuss, and T. Villmann (2007), Magnification control for batch neural gas, *Neurocomputing* 70:1225-1234.

[10] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert (2004), Recursive self-organizing network models, *Neural Networks* 17(8-9):1061-1086.

[11] B. Haasdonk and C. Bahlmann (2004), Learning with distance substitution kernels, in *Pattern Recognition - Proc. of the 26th DAGM Symposium*.

[12] R. J. Hathaway and J. C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition* 27(3):429-437, 1994.

[13] R. J. Hathaway, J. W. Davenport, and J. C. Bezdek. Relational duals of the c-means algorithms. *Pattern Recognition* 22:205-212, 1989.

[14] T. Heskes (2001). Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks* 12:1299-1305.

[15] T. Hofmann and J. Buhmann Multidimensional Scaling and Data Clustering (1995), in *Advances in Neural Information Processing Systems* 7:459-466.

[16] A. Juan and E. Vidal (2000), On the use of normalized edit distances and an efficient k-NN search technique (k-AESA) for fast and accurate string classification, in *ICPR 2000*, vol.2, p. 680-683.

[17] S. Kaski, J. Nikkilä, E. Savia, and C. Roos (2005), Discriminative clustering of yeast stress response, In *Bioinformatics using Computational Intelligence Paradigms*, U. Seiffert, L. Jain, and P. Schweizer (eds.), pages 75-92, Springer.

[18] S. Kaski, J. Nikkilä, M. Oja, J. Venna, P. Törönen, and E. Castren (2003), Trustworthiness and metrics in visualizing similarity of gene expression, *BMC Bioinformatics*, 4:48.

[19] T. Kohonen (1995), *Self-Organizing Maps*, Springer.

[20] T. Kohonen and P. Somervuo (2002), How to make large self-organizing maps for nonvectorial data, *Neural Networks* 15:945-952.

[21] J. B. Kruskal (1964), Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29:1-27.

[22] C. Lundsteen, J. Phillip, and E. Granum (1980), Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes, *Clinical Genetics* **18**:355-370.

[23] T. Martinetz, S.G. Berkovich, and K.J. Schulten (1993), 'Neural-gas' network for vector quantization and its application to time-series prediction, *IEEE Transactions on Neural Networks* 4:558-569.

[24] T. Martinetz and K. Schulten (1994), Topology representing networks. *Neural Networks* 7:507-522.

[25] H. Mevissen and M. Vingron (1996), Quantifying the local reliability of a sequence alignment, *Protein Engineering* **9:**127-132.

[26] M. Neuhaus and H. Bunke (2006), Edit distance based kernel functions for structural pattern classification *Pattern Recognition* 39(10):1852-1863.

[27] Neural Networks Research Centre, Helsinki University of Technology, SOM Toolbox, http://www.cis.hut.fi/projects/somtoolbox/

[28] A.K. Qin and P.N. Suganthan (2004), Kernel neural gas algorithms with application to cluster analysis, *ICPR 2004* vol.4, pp.617-620.

[29] H. Ritter (1999), Self-organizing Maps in non-euclidean Spaces, *Kohonen Maps*, 97-108, Eds.: E. Oja and S. Kaski.

[30] F. Rossi, A. Hasenfuss, B. Hammer, Accelerating relational clustering algorithms with sparse prototype representation, submitted to WSOM'07.

[31] A. Saalbach, T. Twellmann, A. Wismüller, J. Ontrup, H. Ritter, T. W. Nattkemper (2005), A Hyperbolic Topographic Mapping for Proximity Data, *Proc. of the IASTED International Conference on Artifical Intelligence and Applications*, 106-111, ACTA Press.

[32] J. W. Sammon Jr. (1969), A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers* 18:401-409.

[33] F.-M. Schleif, B. Hammer, and T. Villmann (2007), Margin based Active Learning for LVQ Networks, *Neurocomputing* 70(7-9):1215-1224.

[34] S. Seo and K. Obermayer (2004), Self-organizing maps and clustering methods for matrix data, *Neural Networks* 17:1211-1230.

[35] M. Strickert, U. Seiffert, N. Sreenivasulu, W. Weschke, T. Villmann, B. Hammer (2006), Generalized Relevance LVQ (GRLVQ) with Correlation Measures for Gene Expression Analysis, *Neurocomputing 69*: 651-659.

[36] P. Tino, A. Kaban, and Y. Sun (2004), A generative probabilistic approach to visualizing sets of symbolic sequences. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD-2004*, (eds) R. Kohavi, J. Gehrke, W. DuMouchel, J. Ghosh. pp. 701-706, ACM Press.

[37] T. Villmann, B. Hammer, F. Schleif, T. Geweniger, and W. Herrmann (2006), Fuzzy classification by fuzzy labeled neural gas, *Neural Networks*, 19:772-779.

[38] J. Walter (2004), H-MDS: a new approach for interactive visualization with multidimensional scaling in the hyperbolic space. *Information Systems*, 29(4):273-292.

[39] W.H. Wolberg, W.N. Street, D.M. Heisey, and O.L. Mangasarian (1995), Computer-derived nuclear features distinguish malignant from benign breast cytology, *Human Pathology*, **26**:792–796.

[40] H. Yin (2006), On the equivalence between kernel self-organising maps and self-organising mixture density network, *Neural Networks* 19(6):780-784.